Outline

- Set, Relations, and Functions
- Partial Orders
- Boolean Functions
- Don’t Care Conditions
- Incomplete Specifications
Set Notation

\( v \in V \)  Element \( v \) is a member of set \( V \)

\( v \notin V \)  Element \( v \) is not a member of set \( V \)

\(|V|\)  Cardinality (number of members) of set \( V \)

\( S \subseteq V \)  Set \( S \) is a subset of \( V \)

\( \emptyset \)  The empty set (a member of all sets)

\( S' \)  The complement of set \( S \)

\( U \)  The universe: \( S' = U - S \)
Set Notation

- Inclusion ($\subseteq$)
- Proper Inclusion ($\subset$)
- Complementation
- Intersection ($\cap$)
- Union ($\cup$)
- Difference
Power Sets

\[ 2^V = \{ S \mid S \subseteq V \} \]

2^V

The power set of set V (the set of all subsets of set V)

|2^V| = 2^{|V|}

The cardinality of a power set is a power of 2
Power Sets

\[ V = \{0,1,2\} \]  
3-member set

\[ 2^V = \{\varnothing, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\} \]

1 subset with 0 members
3 subsets with 1 members
3 subsets with 2 members
1 subset with 3 members

\[ |2^V| = 2^{|V|} = 2^3 = 8 \]

Power sets are Boolean Algebras
Cartesian Products

The **Cartesian Product** of sets A and B is denoted $A \times B$

Suppose $A = \{0,1,2\}$, $B = \{a,b\}$, then

$$A \times B = \{(0,a),(0,b),(1,a),(1,b),(2,a),(2,b)\}$$

$A = \{0,1,2\}$ Set A is unordered

$(1,b)$ $(\ )$ denotes Ordered Set
Binary Relations

- The **Cartesian Product** of sets $A$ and $B$ is denoted $A \times B$
- $A \times B$ consists of all possible ordered pairs $(a,b)$ such that $a \in A$ and $b \in B$
- A subset $R \subseteq A \times B$ is called a **Binary Relation**
- Graphs, Matrices, and Boolean Algebras can be viewed as binary relations
Binary Relations as Graphs or Matrices

\[ E = \{ab, cb, cd, cf\} \subseteq A \times B \]

\[ A = \{a, c\} \quad B = \{b, d, f\} \]

**Rectangular Matrix**

\[
\begin{array}{ccc}
& b & d & f \\
\hline
a & 1 & 0 & 0 \\
c & 1 & 1 & 1 \\
\end{array}
\]

**Bipartite Graph**

**Rectangular Matrix**
Binary Relations as Graphs or Matrices

\[ E = \{ab, ac, ad, bd, cd\} \cup \{aa, bb, cc, dd\} \subseteq V \times V \]

\[ V = \{a, b, c, d\} \]

Directed Graph

Square Matrix

\[
\begin{array}{cccc}
a & 1 & 1 & 1 \\
b & 0 & 1 & 0 \\
c & 0 & 0 & 1 \\
d & 0 & 0 & 0 \\
\end{array}
\]
Properties of Binary Relations

• A binary relation $R \subseteq V \times V$ can be
  ▪ reflexive, and/or
  ▪ transitive, and/or
  ▪ symmetric, and/or
  ▪ antisymmetric

• We illustrate these properties on the next few slides
Notation for Binary Relations

If \( R \subseteq A \times B \), we say that \( A \) is the **domain** of the relation \( R \), and that \( B \) is the **range**.

If \( (a, b) \in R \), we say that the pair is in the relation \( R \), or \( aRb \).
\[ \leq \subseteq A^2 = \{1,2,3,4\}^2 \]

\[
\begin{array}{cccc}
1 & \leq & \leq & \leq \\
\leq & \leq & \leq & \leq \\
\leq & \leq & \leq & \\
\leq & & & \\
\end{array}
\]

Graph View
Example: “Less than or Equal”

Set and Relations

\[ A = B = \{0, 1, 2, \ldots \}, \]
\[ R \subseteq A \times B = "\leq" \]

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & \leq & \leq & \leq \\
1 & 0 & \leq & \leq \\
2 & 0 & 0 & \leq \\
3 & \vdots & \vdots & \vdots \\
\end{array}
\]
Example: “$a$ times $b = 12$”

$V = \{0,1,2,\ldots\}$,

$R \subseteq V \times V = \{(u,v) | u \times v = 12\}$

$R = \{(1,12), (2,6), (3,4), (4,3), (6,2), (12,1)\}$
Reflexive Binary Relations

A binary relation $R \subseteq V \times V$ is reflexive if and only if $(v, v) \in R$ for every vertex $v \in V$. 

$$V = \{a, b, c, d\}$$

$$a \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$b \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$$

$$c \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$$

$$d \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$
Non-Reflexive Binary Relations

\[ R \subseteq V \times V \]
\[ \exists v \in V \ni \neg vRv \]

Non-Reflexivity implies that there exists \( v \in V \) such that \((v, v) \notin R\). Here \( d \) is such a \( v \).

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]
Transitive Binary Relations

If \( u, v, w \in V \), and \( uRv \), and \( vRw \), then \( uRw \).

A binary relation is transitive if and only if every \((u, v, w)\) path is triangulated by a direct \((u, w)\) edge.

This is the case here, so \( R \) is transitive.
Non-Transitive Binary Relations

A binary relation is not transitive if there exists a path from $u$ to $w$, through $v$ that is not triangulated by a direct $(u, w)$ edge.

Here $(a, c, e)$ is such a path.
A binary relation is symmetric if and only if every \((u, v)\) edge is reciprocated by a \((v, u)\) edge
A binary relation is non-symmetric if there exists an edge $(u, v)$ not reciprocated by an edge $(v, u)$.

\[ R \subseteq V \times V \]

\[ \exists (u, v) \in R \text{ such that } (v, u) \notin R \]
Antisymmetric Binary Relations

\( \forall (u, v) \in R, (uRv, vRu) \Rightarrow (v = u) \)

A binary relation is antisymmetric if and only if no \((u,v)\) edge is reciprocated by a \((v,u)\) edge \(v = u\)

Not antisymmetric if any such edge is reciprocated:
here \( b \rightarrow d, d \rightarrow b \)
Functions

- A function $f$ are binary relations from set $A$ (called the domain) to $B$ (called the range).
- But, it is required that each $a$ in $A$ be associated with exactly 1 $b$ in $B$.
- For functions, it cannot be true that both $(a,b)$ in $R$ and $(a,c)$ in $R$, $b$ different from $c$. 
The Binary Relation of Relations to Synthesis/Verification

- **Partial Orders**
- **Equivalence Relations**
- **Compatibility Relations**

- **Combinational Logic (no DCs)**
  - $\Rightarrow$ (0,1) Boolean Algebra
- **Combinational Logic (with DCs)**
  - $\Rightarrow$ Big Boolean Algebras
- **Sequential Logic (no DCs)**
- **Sequential Logic (with DCs)**

*DC=don’t care*
The Path Relation

\[ G = (V, E) \]

This graph defines path relation

Relation \( \rightarrow^* \) is sometimes called “Reachability”
Note $R$ is reflexive, symmetric, and transitive.
Equivalence Relations

This graph defines path relation $\rightarrow^*$

$G=(V,E)$

$R=\{(u,v) | v \rightarrow^* u, u \rightarrow^* v\}$

$G=(V,R)$

This graph defines $R$

NOT an equivalence relation: $E$

An equivalence relation: $R$
The Cycle Relation

Given a graph $G=(V,E)$, the cycle relation $C$ defines a set of pairs $(u,v)$ such that $v \rightarrow^* u$. These pairs form the strongly connected components of $G$.

This graph defines path relation $\rightarrow^*$.

These are called the “Strongly Connected Components” of $G$. 
Refinement

• Given any set B a **partition** of B is a set of subsets $B_i \subseteq B$ with two properties
  - $B_i \cap B_j = \emptyset$ for all $i \neq j$
  - $\bigcup_i B_i = B$

• Given two partitions $P^1$ and $P^2$ of a set S, $P^1$ is a **refinement** of $P^2$ if each block $B^1_i$ of $P^1$ is a subset of some block of $P^2$
Other Binary Relations

- Partial Orders (Includes Lattices, Boolean Algebras)
  - Reflexive
  - Transitive
  - Antisymmetric

- Compatibility Relations
  - Reflexive
  - Not Transitive—Almost an equivalence relation
  - Symmetric
Functions

- A **function** $f$ from $A$ to $B$ written $f : A \to B$ is a rule that associates exactly one element of $B$ to each element of $A$
  - A relation from $A$ to $B$ is a function if it is right-unique and if every element of $A$ appears in one pair of the relation
  - $A$ is called the domain of the function
  - $B$ is called the co-domain (range)
- If $y=f(x) : A \to B$, $y$ is **image** of $x$
  - Given a domain subset $C \subseteq A$
    - $\text{IMG}(f,C) = \{ y \in B \mid \exists x \in C \ni y = f(x) \}$
    - **preimage** of $C$ under $f$
      - $\text{PRE}(f,C) = \{ x \in A \mid \exists y \in C \ni y = f(x) \}$
- A function $f$ is one-to-one (injective) if $x \neq y$ implies $f(x) \neq f(y)$
- A function $f$ is onto (surjective) if for every $y \in B$, there exists an element $x \in A$, such that $f(x) = y$
The pair \((V, \leq)\) is called an **algebraic system**

- \(V\) is a set, called the **carrier** of the system
- \(\leq\) is a relation on \(V \times V = V^2\)
  
  (\(\subseteq\), \(\implies\) are similar to \(\leq\))

- This algebraic system is called a **partially ordered set**, or **poset** for short
Partially Ordered Sets

- A poset has two operations, $\bullet$ and $+$, called meet and join (like AND and OR)
- Sometimes written $(V, \leq, \cdot, +)$ or $(V^2, \leq, \cdot, +)$ even $(V, \cdot, +)$, since $\leq$ is implied
Integers (Totally Ordered):

\[ \leq \subseteq A^2 = \{1,2,3,4\}^2 \]

Matrix View

Graph View

Hasse Diagram

Hasse Diagram obtained by deleting arrowheads and redundant edges
Posets and Hasse Diagrams

\[(V, \leq, \cdot, +)\]

\[V = \{a, b, c, d\}\]

\[\leq = \{(a, a), (a, b), (a, c), (a, d),\]
\[(b, b), (b, d),\]
\[(c, c), (c, d), (d, d)\}\]

\[\leq: \text{distance from top}\]

\[\leq\] is reflexive, antisymmetric, and transitive: a \textbf{partial order}
An element $m$ of a poset $P$ is a **lower bound** of elements $a$ and $b$ of $P$, if $m \leq a$ and $m \leq b$.

$m$ is the **greatest lower bound** or **meet** of elements $a$ and $b$ if $m$ is a lower bound of $a$ and $b$ and, for any $m'$ such that $m' \leq a$ and also $m' \leq b$, $m' \leq m$. 

[Diagram showing the relationship between $a$, $b$, $m$, and $m'$ in a poset.]
An element $m$ of a poset $P$ is a **upper bound** of elements $a$ and $b$ of $P$, if $a \leq m$ and $b \leq m$.

$m$ is the **least upper bound** or **join** of elements $a$ and $b$ if $m$ is an upper bound of $a$ and $b$ and, for any $m'$ such that $a \leq m'$ and also $b \leq m'$, $m \leq m'$. 
Meet (·) and Join (+)

posets:

\[ a \cdot b = a \]
\[ a + b = b \]

\[ c \cdot b = a \]
\[ c + b = ? \]

\[ c \cdot b = a \]
\[ c + b = d \]

\[ c \cdot a = ? \]
\[ d + b = ? \]
**Theorem 3.2.1** If $x$ and $y$ have a greatest lower bound (meet), then

$$x \geq x \cdot y$$

Similarly, if $x$ and $y$ have a least upper bound (join), then

$$x \leq x + y$$

**Proof:** Since meet exists, $x \geq x \cdot y$ by definition. Also, since join exists, $x \leq x + y$ by definition.
More POSET Properties

Theorem 3.2.2 \( x \leq y \iff x \cdot y = x \)

Proof (\(\Rightarrow\)): This means assume \( x \leq y \).
- \( x \) is a lower bound of \( x \) and \( y \) (by Def.)
- \( x \) is also the meet of \( x \) and \( y \)

Proof: by contradiction. Suppose \( x \neq x \cdot y \). Then \( \exists m \neq x \) such that \( x \leq m \) where \( m = x \cdot y \). But since \( m \) was a lower bound of \( x \) and \( y \), \( m \leq x \) as well. Thus \( m = x \), by the anti-symmetry of posets.

Proof (\(\Leftarrow\)): From Definition of meet
Well-Ordered

- If all pairs of elements of a poset are comparable, then the set is **totally ordered**
- If every non-empty subset of a totally ordered set has a smallest element, then the set is **well-ordered**
  - e.g.) Natural numbers

**Mathematical Induction**

- Given, for all \( n \in \mathbb{N} \), propositions \( P(n) \), if
  - \( P(0) \) is true
  - for all \( n>0 \), if \( P(n-1) \) is true then \( P(n) \) is true
- then, for all \( n \in \mathbb{N} \), \( P(n) \) is true
Lattices

- **Lattice**: a poset with both meet and join for every pair of elements of the carrier set
- **Boolean Algebra**: a distributed and complemented lattice
- Every lattice has a unique minimum element and a unique maximum element
Lattices and Not Lattices

Partial Orders

\[ a \cdot b = a \]
\[ a + b = b \] (lattice)

\[ c \cdot b = a \]
\[ c + b = ? \] (lattice)

\[ c \cdot a = ? \]

\[ d + b = ? \] (Boolean algebra)
Examples of Lattices

\[ d = 1 \]

\[ b \quad c \]
\[ a = 0 \]

\[ g = 1 \]

\[ e \quad f \]
\[ d \]
\[ b \quad c \]
\[ a = 0 \]

\[ e = 1 \]

\[ b \quad c \quad d \]
\[ a = 0 \]
More Notation

- "There Exists a v in set V" is denoted by $\exists v \in V$

- The following are equivalent:

  $a \iff b$ \quad (ab + a'b')

  $a \implies b$ and $a \iff b$ \quad ((a' + b)(a + b'))

  $a$ is true if and only if $b$ is true

- Does $(a' + b)(a + b')$ make sense in a poset?

  No--Complement is defined for lattices but not for posets
Properties of Lattices

Meet, Join, Unique maximum (1), minimum (0) element are always defined

Idempotent: \( x + x = x \) \( x \cdot x = x \)
Commutative: \( x + y = y + x \) \( x \cdot y = y \cdot x \)
Associative: \( x + (y + z) = (x + y) + z \) \( x \cdot (y \cdot z) = (x \cdot y) \cdot z \)
Absorptive: \( x \cdot (x + y) = x \) \( x + (x \cdot y) = x \)

Absorptive properties are fundamental to optimization
Duality

Every lattice identity is transformed into another identity by interchanging:

- $+$ and $\cdot$
- $\leq$ and $\geq$
- $0$ and $1$

Example: $x \cdot (x + y) = x \rightarrow x + (x \cdot y) = x$
Complementation

• If $x+y=1$ and $xy=0$ then $x$ is the complement of $y$ and vice versa
• A lattice is complemented if all elements have a complement
Examples of Lattices

Complemented?

yes

d = 1

b
a = 0

bcd

b · c = a = 0
b + c = d = 1

no

g = 1

e
f
d

b

b · c = a = 0
b + c = d ≠ 1

yes

e = 1

b
c

d

b · c = a = 0
b + c = e = 1

Partial Orders
Distributivity

**Semi-distributivity:**

\[ x \cdot (y + z) \geq (x \cdot y) + (x \cdot z) \]

\[ x + (y \cdot z) \leq (x + y) \cdot (x + z) \]

**Proof:**

1. \( x \cdot y \leq x \) (def. of meet)
2. \( x \cdot y \leq y \leq y + z \) (def. of meet, join)
3. \( x \cdot y \leq x \cdot (y + z) \) (def. of meet)
4. \( x \cdot z \leq x \cdot (y + z) \) (mutatis mutandis: \( y \leftrightarrow z \))
5. \( x \cdot (y + z) \geq (x \cdot y) + (x \cdot z) \) (def. of join)
Distributivity

- **Boolean Algebras** have **full** distributivity:
  \[ x \cdot (y + z) \equiv (x \cdot y) + (x \cdot z) \]
  \[ x + (y \cdot z) \equiv (x + y) \cdot (x + z) \]

- **Boolean Algebras** are complemented. That is,
  \[ x = y' \Rightarrow (x \cdot y = 0) \text{ and } (x + y = 1) \]

must hold for every \( x \) in the carrier of the poset.
Definition of Boolean Algebra

• A complemented, distributive lattice is a Boolean lattice or Boolean algebra
  - Idempotent  \( x+x=x \)  \( xx=x \)
  - Commutative  \( x+y=y+x \)  \( xy = yx \)
  - Associative  \( x+(y+z)=(x+y)+z \)  \( x(yz) = (xy)z \)
  - Absorptive  \( x(x+y)=x \)  \( x+(xy) = x \)
  - Distributive  \( x+(yz)=(x+y)(x+z) \)  \( x(y+z) = xy +xz \)
  - Existence of the complement
Are these Lattices Boolean Algebras?

Complemented and distributed?

Yes

\[ d = 1 \]

\[ b \quad c \]

\[ a = 0 \]

\[ b \cdot c = a = 0 \]
\[ b + c = d = 1 \]
\[ a(b + c) = a = 0 \]
\[ a \cdot b + a \cdot c = a = 0 \]

No

\[ g = 1 \]

\[ e \quad f \]

\[ d \]

\[ b \quad c \]

\[ a = 0 \]

\[ b \cdot c = a = 0 \]

\[ b + c = d \neq 1 \]

No

\[ e = 1 \]

\[ b \quad c \quad d \]

\[ a = 0 \]

\[ b \cdot (c + d) = b \geq 1 \]

\[ (b \cdot c) + (b \cdot d) = a \]
Fundamental Theorem of Boolean Algebras

- Every poset which is a Boolean Algebra has a power of 2 elements in its carrier.

- All Boolean Algebras are isomorphic to the power set of the carrier.

Example:

\[ V = \{1\}, \quad 2^V = \{\emptyset, 1\} = \{0, 1\} \]
Examples of Boolean Algebras

\[ V = \{1\}, \quad 2^V = \{\emptyset, 1\} = \{0, 1\} \]

\[ V = \{0, 1\}, \quad 2^V = \{\emptyset, 0, 1, \{0, 1\}\} \]
\[ = \{0, 0, 1, 1\} \]
Atoms of a Boolean Algebra

- A Boolean Algebra is a Distributive, Complemented Lattice
- The minimal non-zero elements of a Boolean Algebra are called “atoms”

\[ |V| = 2^n \iff n \text{ atoms} \]

1-atom

0

2-atoms

0

1

Can a Boolean Algebra have 0 atoms? **NO!**
Examples of Boolean Algebras

\[ A = \{a, b, c\} \]
\[ V = 2^A \]
\[ = \{A, \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\} \]

\[ a, b, c, \quad \text{3 atoms} \]
\[ \emptyset \]
Examples of Boolean Algebras

\[ n = |A| = 4 \text{ atoms, } n+1 = 5 \text{ levels, } 2^n = 16 \text{ elts} \]

\[ A = \{a, b, c, d\} \]

\[ V = 2^A \]

Level 4--1 elt* = \{A, \}

Level 3--4 elts \{a, b, c\}, \{a, b, d\}, \{a, c, d\}, \{b, c, d\}, \]

Level 2--6 elts \{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\} \]

Level 1--4 elts \{a\}, \{b\}, \{c\}, \{d\} \]

Level 0--1 elts \emptyset \]

* “elt” = element

\[ C_4 \]

\[ C_3 \]

\[ C_2 \]

\[ C_1 \]

\[ C_0 \]
Theorem 3.2.6  Complementation is unique.

**proof:** Suppose \( x' \) and \( y \) are both complements of \( x \) \((x + y = 1, xy = 0)\). Hence

\[
y = y(x + x') = x'y + xy = x'y
\]

\[
= x'y + x'x = x'(y + x) = x'
\]

Note we used distributivity. Similarly, we have

**Theorem 3.2.7 (Involution):** \((x')' = x\)
Properties of Boolean Algebra

- \( x + x'y = x + y \)
- \( x(x' + y) = xy \)
Lemma: The Isotone Property

Proof: By Theorem 3.2.2, $x \leq y \iff x = xy$, so we get $xz = xy = xyz = (xz)(yz)$. Note we used idempotence, commutativity, and associativity. Then we use Theorem 3.2.2 again to prove the lemma.
Duality for Boolean Algebras

Every Boolean Algebra identity is transformed into another valid identity by interchanging:

- $+$ and $\cdot$
- $\leq$ and $\geq$
- $0$ and $1$
- $()'$ and $()$  This rule not valid for lattices

Example: $xx' = 0 \rightarrow x' + x = 1$
Theorem

Proof: By the isotone property we have

\[ x \leq y \iff xy' = 0 \]
\[ \iff x' + y = 1 \]

The second identity follows by duality.
DeMorgan’s Laws

\[(x + y)′ = x′y′\]
\[(xy)′ = x′ + y′\]

Consensus

\[xy + x′z + yz = xy + x′z\]
\[(x + y)(x′ + z)(y + z) = (x + y)(x′ + z)\]
Consensus Examples

1. $a'bc + abd + bcd = abc + a'bd$

3. $abe + bce + bde + ac'd'$
   
   $= abe + be(c + d) + a(c + d)'$
   
   $= be(c + d) + a(c + d)'$
   
   $= bce + bde + ac'd'$

Note use of DeMorgan’s Law
Consensus Example

5. Is wrong. Replace by

\[ a'c'd + b'c'd + acd + bcd \]

\[ = (a'bd) + a'c'd + b'c'd + acd + bcd \]

\[ = a'bd + b'c'd + acd \]

• Uphill Move: Note addition of redundant consensus term enables deletion of two other terms by consensus
• This avoids local minima—a crucial part of the logic minimization paradigm
Ordinary \textbf{functions} of 1 variable:

\[ f(x) : D \mapsto R \iff f \subseteq D \times R \]

Ordinary \textbf{functions} of 2 real variables:

\[ f(x, y) : D_x \times D_y \mapsto R \iff f \subseteq (D_x \times D_y) \times R \]
Boolean functions of $n$ variables:

$$f(x_1,\ldots,x_n): B^n \mapsto B,$$

$$B = \{0,\ldots,1\}$$

$$f(x_1,\ldots,x_n) \subseteq (B \times \cdots \times B) \times B$$
Boolean Formulae: Meets and Joins of Variables and Constants

\[ F_1 = 0 \]
\[ F_2 = x_1 x'_2 + x'_1 x_2 \]
\[ F_3 = x_1 x_1 x'_2 + x'_1 x_2 \]
\[ F_3 = a x_1 + b \]
n-variable Boolean Formulae

\[ B = \{0, \ldots, 1\} \]

- The elements of \( B \) are Boolean formulae.
- Each variable \( x_1, \ldots, x_n \) is a Boolean formula.
- If \( g \) and \( h \) are Boolean formulae, then so are
  - \( g + h \)
  - \( g \cdot h \)
  - \( g' \)
- A string is a Boolean formula if and only if it derives from finitely many applications of these rules.
Distinct Formulas, Same Function

$$B = \{0, a, b, 1\}$$

Truth Table for $$f: B^2 \rightarrow B$$

<table>
<thead>
<tr>
<th>$$x_1$$</th>
<th>$$x_2$$</th>
<th>0000</th>
<th>aaaa</th>
<th>bbbb</th>
<th>1111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0ab1</td>
<td>0ab1</td>
<td>0ab1</td>
<td>0ab1</td>
<td>0ab1</td>
<td></td>
</tr>
<tr>
<td>0ab1</td>
<td>aa11</td>
<td>b1b1</td>
<td>1111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$F = x_1 + x_2$$

$$G = x_1 + x_1'x_2$$
Boolean Functions on Boolean Algebra

\[ B = \{0, \ldots, 1\} \]

- \( f(x_1, \ldots, x_n) = x_i \) is a Boolean function
- \( f(x_1, \ldots, x_n) = e \in B \) is also
- If \( g \) and \( h \) are Boolean, functions then so are
  - \( g + h \)
  - \( g \cdot h \)
  - \( g' \)
- A function is Boolean if and only if it derives from finitely many applications of these rules.
Cofactors

Positive Cofactor WRT $x_1$: $f_{x_1} = f(1, x_2, \ldots, x_n)$

Negative Cofactor WRT $x_1$: $f_{x_1'} = f(0, x_2, \ldots, x_n)$

$Note$: prime denotes complement

$$f_{x_1} = f(1, x_2, \ldots, x_n) = f_{x_1=1}$$

$$f_{x_1'} = f(0, x_2, \ldots, x_n) = f_{x_1=0}$$
Cofactors

Positive Cofactor WRT $x_1$: $f_{x_1} = f(1, x_2, \ldots, x_n)$

Negative Cofactor WRT $x_1$: $f_{x_1}' = f(0, x_2, \ldots, x_n)$

Example:

$$f = abc'd + a'cd' + bc$$

$$f_a = bc'd + bc$$

$$f_{a'} = + cd' + bc$$

This term drops out when $a$ is replaced by $1$

This term is unaffected
Cofactors

\[ B = \{0, a, b, 1\}, \quad f(x): B^n \mapsto B \]

\[ f_{x_1} = f(a, x_2, \ldots, x_n) \]

\[ f_{x_1} = f(1, x_2, \ldots, x_n) \]

Example:

\[ f = ax'_1 + bx_2 \]

\[ f_{x_1} = aa' + bx_2 = 0 + bx_2 = bx_2 \]
Boole’s Expansion Theorem

\[ f(x_1, x_2, \ldots, x_n) = \]

\[ x_i f_{x_i} + x'_i f_{x'_i} = [x_i + f_{x'_i}] \cdot [x'_i + f_{x_i}] \]

Example: \( f = ax'_1 + bx_2, \)

\[ f_{x_1} = a0 + bx_2 = bx_2, \quad f_{x'_1} = a1 + bx_2 = a + bx_2 \]

Sum form: \( f = x_1(bx_2) + x'_1(a + bx_2) \)

Product form: \( f = [x_1 + (a + bx_2)] \cdot [x'_1 + (bx_2)] \)
Variables and Constants

In the previous slide $a$ and $b$ were constants, and the $x_1, \ldots, x_n$ were variables.

But we can also use letters like $a$ and $b$ as variables, without explicitly stating what the elements of the Boolean Algebra are.
Boole’s Expansion (Sum Form)

\[ f = abc'd + a'cd' + bc \]

\[ f_a = bc'd' + bc \]

\[ f_{a'} = + cd' + bc \]

\[ f = af_a + a'f_{a'} = a(bc'd + bc) + a'(cd' + bc) \]
Boole’s Expansion (Product Form)

\[ f = abc'd + a'cd' + bc \]

\[ f = [a + f_a'][a' + f_a] \]

\[ = [a + (cd' + bc)][a' + (bc'd + bc)] \]

\[ = aa' + a(bc'd + bc) + a'(cd' + bc) + (cd' + bc)(bc'd + bc) \]

\[ = abc'd + a'cd' + bcd' + bc \]

\[ = abc'd + a'cd' + bc \]

Note application of absorptive law
The minterm canonical form is a canonical, or standard way of representing functions. From page 10

\[ f = x + y' + \text{is represented by millions of distinct Boolean formulas, but just 1 minterm canonical form. Note} \]

\[ f = x + y' + z = (x'yz')' \]

Thus some texts refer to \( f \) as \{0, 1, 3, 4, 5, 6, 7\}

\[ f = x'y'z' + x'y'z + x'yz + xy'z + \cdots \]

\[(0, 1, 3, 4, \cdots)\]
Minterm Canonical Form

\[ f(x_1, x_2, \ldots, x_n) = x'_1 f_{x'_1} + x_1 f_{x_1} \]

\[ x'_1 x'_2 f_{x'_1 x'_2} + x'_1 x_2 f_{x'_1 x_2} + x_1 x'_2 f_{x_1 x'_2} + x_1 x_2 f_{x_1 x_2} \]

... 

\[ = x'_1 \cdots x'_n f_{x'_1 \cdots x'_n} + x_1 \cdots x_{n-1} x_n f_{x_1 \cdots x_{n-1} x_n} + \cdots \]

\[ + x'_1 x_2 \cdots x_n f_{x'_1 x_2 \cdots x_n} + x_1 \cdots x_n f_{x_1 \cdots x_n} \]

These elementary functions are called minterms.
Minterm Canonical Form

Thus a Boolean function is uniquely determined by its values at the corner points 0⋯0, 0⋯1, ⋯, 1⋯1

\[ f(x_1, x_2, \ldots, x_n) = x'_1 f_{x_1'} + x_1 f_{x_1} \]
\[ = x'_1 x'_2 f_{x'_1 x'_2} + x'_1 x_2 f_{x'_1 x_2} + x_1 x'_2 f_{x_1 x'_2} + x_1 x_2 f_{x_1 x_2} \]
\[ \quad \vdots \]
\[ = x'_1 \cdots x'_n f_{x'_1 \cdots x'_n} + x'_1 \cdots x'_{n-1} x_n f_{x'_1 \cdots x'_{n-1} x_n} + \cdots \]
\[ + x'_1 x_2 \cdots x_n f_{x'_1 x_2 \cdots x_n} + x_1 \cdots x_n f_{x_1 \cdots x_n} \]

These \(2^n\) constants are aptly called **discriminants**.
Minterm Canonical Form

\[ f : B^2 \rightarrow B \quad B = \{0, a, b, 1\} \]

\[ f = ax'_1 + bx_2, \quad f_{x'_1} = a + bx_2, \quad f_{x_1} = bx_2 \]

\[ = x'_1x'_2a + x'_1x_2(a + b) + x_1x'_20 + x_1x_2b \]

This function is Boolean (from Boolean Formula)

\[ F = ax'_1 + bx_2 \]

\[ x'_1x'_2a + x'_1x_2 + x_1x_2b \]

Thus all 16 cofactors match--not just discriminants
Now Suppose Truth Table is Given

- Here we change 15th cofactor, but leave the 4 discriminants unchanged
- Since the given function doesn’t match at all 16 cofactors, $F$ is **not Boolean**

\[
x'_1x'_2a + x'_1x_2 + x_1x_2b
\]

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0ab1</td>
<td>0ab1</td>
</tr>
<tr>
<td>0ab1</td>
<td>00bb</td>
<td>a11</td>
</tr>
<tr>
<td>a11</td>
<td>00bb</td>
<td>a11</td>
</tr>
<tr>
<td>a11</td>
<td>00bb</td>
<td>a11</td>
</tr>
</tbody>
</table>

Since the given function doesn’t match at all 16 cofactors, $F$ is **not Boolean**.
2-Variable Boolean Functions

Minterms 2-variable functions

<table>
<thead>
<tr>
<th>4</th>
<th>$f^{15}$ = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$f^{11-14}$ = $x' + y'$, $x' + y$, $x + y'$, $x + y$</td>
</tr>
<tr>
<td>2</td>
<td>$f^{5-10}$ = $x'$, $y'$, $x$, $y$, $xy' + x'y$, $xy + x'y'$</td>
</tr>
<tr>
<td>1</td>
<td>$f^{1-4}$ = $x'y'$, $xy'$, $x'y$, $xy$</td>
</tr>
<tr>
<td>0</td>
<td>$f^{0}$ = 0</td>
</tr>
</tbody>
</table>

Note that the minterms just depend on the number and names of the variables, independent of the particular Boolean Algebra.
The notation $F_n(B)$ means “the Boolean Algebra whose carrier is the set of all n-variable Boolean Functions” which map $B$ into $F_n(B)$: $B^n \rightarrow B$ minterms

$$F_n(B): B^n \rightarrow B$$

If $B = \{0, 1\}$ the atoms of $F_n(B)$ are its n-variable

$B$ is called the “Base Algebra” of the Boolean Function algebra

The atoms of $B$ are called Base Atoms
Large Boolean Algebras?

- When you design an optimal circuit, each gate must be optimized with respect to its Don’t Cares.
- Because of Don’t Cares, 4 functions of \((x, y)\) are equivalence preserving replacements for gate \(g\).
- Optimal Design: pick best such replacement.
Boolean Algebra

\[ F_1(\{0, a, b, 1\}) \]

Level (= # of atoms)

4

Each element has 4 Atoms

3  \( x' + a \) \( x + a \) \( x + b \) \( x' + b \)

Each element has 3 Atoms

2  \( a \) \( xa + x'b \) \( x \) \( x' \) \( xb + x'a \) \( b \)

Each element has 2 Atoms

1  \( xa \) \( x'a \) \( xb \) \( xb \)

Each element has 1 Atom

Note: base atoms act like 2 extra literals \( a \approx y \), \( a' = b \approx y' \)
Boolean Algebra

\[ F_2(\{0,1\}) \]

Level (\(\#\) of atoms)

4

3 minterms (3 atoms)

3

2 minterms (2 atoms)

2

1 minterm (1 atom)

1
Boolean Algebra $F_1(\{0,a,b,1\})$ of 1-Variable Functions

atoms of $F_1(\{0,a,b,1\})$ are base atoms·minterms

3 Atoms

2 Atoms

1 Atom

Subalgebras are **incompletely specified** Boolean Functions (most important)
Boolean Subalgebra

Boolean Subalgebra: Interval of 2-Variable Function Lattice

The 3-Atom element \( x + y = xy' + xy + x'y \) is **ONE** of subalgebra

2-Atom elements are atoms of subalgebra

1-Atom element \( xy' \) is **ZERO** of subalgebra
More On Counting

Level (= # of atoms)

4

An algebra (or subalgebra) with \( n + 1 \) levels has exactly \( 2^n \) elements, because \( n + 1 \) levels implies \( n \) atoms
A larger Subalgebra

Interval of 2-Variable Function Lattice

ONE of algebra (4-Atoms) is also one of this subalgebra

2-Atom elements are atoms of subalgebra

1-Atom element is ZERO of subalgebra
Counting Elements of $F_n(B)$

- For each of the $2^n$ minterms, the discriminant can be chosen as any of the $|B|$ elements of $B$.
- Therefore, the number of elements of $F_n(B)$ is

Examples

\[
|B|^{(2^n)} = (2^{|A(B)|})^{2^n} = 2^{(|A(B)| \cdot 2^n)}
\]

$B = \{0, 1\}, \quad |A(B)| = 1$

\[
n = 2 \implies |B|^{(2^n)} = 2^{2^2} = 2^4 = 16
\]

\[
n = 3 \implies |B|^{(2^n)} = 2^{2^3} = 2^8 = 256
\]
Counting Elements of $F_n(B)$

• Examples with $B = \{0, a, b, 1\}$, $A(B) = \{a, b\}$

\[
n = 2 \Rightarrow 2^{(|A(B)| \cdot 2^n)} = 2^{2 \cdot 2^3} = 2^{16}
\]

\[
n = 3 \Rightarrow 2^{(|A(B)| \cdot 2^n)} = 2^{2 \cdot 2^4} = 2^{32}
\]

• Examples with $|A(B)| = 4$

• Note 2(4) base atoms act like 1(2) extra variables

\[
n = 2 \Rightarrow 2^{(|A(B)| \cdot 2^n)} = 2^{4 \cdot 2^3} = 2^{32}
\]

\[
n = 3 \Rightarrow 2^{(|A(B)| \cdot 2^n)} = 2^{4 \cdot 2^4} = 2^{64}
\]
Boolean Difference (Sensitivity)

A Boolean function $f$ depends on $a$ if and only if $f_a \neq f_{a'}$. Thus

$$\frac{\partial f}{\partial a} = f_a \oplus f_{a'}$$

is called the Boolean Difference, or Sensitivity of $f$ with respect to $a$. 
Example:

\[ f = abc + a'b'c \]

\[ \frac{\partial f}{\partial a} = f_a \oplus f_{a'} = (bc) \oplus (bc) = 0 \]

\[ \frac{\partial f}{\partial b} = f_b \oplus f_{b'} = (ac) \oplus (a'c) = c \]

Note the formula depends on \( a \), but the implied function does not
Don’t Care

- An interval \([L, U]\) in a Boolean algebra \(B\) is the subset of \(B\) defined by \([L,U] = \{ x \in B : L \leq x \leq U \}\)
- Satisfiability don’t cares
- Observability don’t cares
Intervals and Don’t Cares

\[ D = D^{Obs} + D^{Sat} = x'y' \]
\[ L = g - D = gD' \]
\[ = (x'y + xy')(x + y) = (x'y + xy') = g \]
\[ U = g + D = (x'y + xy') + x'y' \]
\[ = x' + y' \]
Don’t Cares

The complete don’t care set for gate \( g \) is

\[
D^g = D^{Sat} + D^{Obs}
\]
Don’t Cares

For this circuit, local input combinations $x'y'$ ($x = 0$, $y = 0$) do not occur. That is, the local minterm $x'y'$ is don’t care.

$x = u' + v'$, $y = v$

$(y = 0) \Rightarrow (x = 1)$
Don’t Cares

For this circuit, global input combination 10 sets

\[ t' = u v' = 1 \]

which makes \( z \) insensitive to \( w \). However, local input pair 10 (\( x y' \)) is **NOT** don’t care, since \( u' v' \)
also gives \( x y' \), and in this case \( t_{u'v'} = 1 \).

\[ z = w' + t' \]

\[ \frac{\partial z}{\partial w} = z_w \oplus z_{w'} \]

\[ = t' \oplus 1 \]

\[ = t \]
Computing ALL Don’t Cares

\[
\begin{array}{cccccccc}
 u & v & x & y & w & t & z & \frac{\partial z}{\partial w} \\
 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

\[xy \in D^{Sat} \text{ if and only if } f^x(u, v) = x \text{ and } f^y(u, v) = y \text{ does not occur for any row } u, v \text{ in the truth table.} \]

Here, \( D^{Sat} = x'y' \) (00 does not occur)
Computing ALL Don’t Cares

Similarly $xy \in D^{Obs}$ if and only for every row $u, v$ such that $f^x(u, v) = x$ and $f^y(u, v) = y$, 

\[
\frac{\partial z}{\partial w} = z_w \oplus z_{w'} = 0
\]

Note these differ!

<table>
<thead>
<tr>
<th>$u$</th>
<th>$v$</th>
<th>$x$</th>
<th>$y$</th>
<th>$w$</th>
<th>$t$</th>
<th>$z$</th>
<th>$\frac{\partial z}{\partial w}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Here 10 ($xy'$) is **NOT** don’t care since $\frac{\partial z}{\partial w} = 1$ in the first row.
Computing ALL Don’t Cares

Don't Cares

Note these differ!

<table>
<thead>
<tr>
<th>u</th>
<th>v</th>
<th>x</th>
<th>y</th>
<th>w</th>
<th>t</th>
<th>z</th>
<th>∂z / ∂w</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus the exclusive OR gate can be replaced by a NAND

\[ D^g = D^{Sat} + D^{Obs} = x'y' \]

\[ g = xy' + x'y (+x'y') \rightarrow x' + y' \]
Suppose we are given a Boolean Function $f$ and a don’t care set $D$. Then any function in the interval (subalgebra)

$$[f_L, f_U] = [fD', f + D]$$

is an acceptable replacement for $f$ in the environment that produced $D$. Here $fD'$ is the 0 of the subalgebra and $f + D$ is the 1.
Suppose we are given a Boolean Function \( g \) and a don’t care set \( D \). Then the triple 
\[(f, d, r)\]
where \( f = gD' \), \( d = D \), and \( r = (f + D)' \) is called an Incompletely specified function.

Note \( f + d + r = gD' + D + (g + D)' = 1 \).